

Forecasting through k-nearest neighbours

Business Intelligence per i Servizi Finanziari 2023-2024

Antonio Candelieri

Time Series Forecasting

- ▶ Given a time-series $X_{t-p:t} = \{x_i\}_{i=t-p,\dots,t}$, with $x_t \in \mathbb{R}^d$, three different tasks can be considered:

- ▶ Forecasting the value(s) of the series **1 step ahead**: $x_{t+1} = f(X_{t-p:t})$
 - ▶ Forecasting the value(s) of the series **at $h > 0$ steps ahead**: $x_{t+h} = f(X_{t-p:t})$
 - ▶ Forecasting the values of the series **on a time window having width h** : $X_{t+1:t+h} = f(X_{t-p:t})$
- } single-step
- } multi-step

▶ Multi-output vs recursive forecast:

- ▶ Multi-output: all the values $X_{t+1:t+h}$ are computed in one-shot
- ▶ Recursive: single-step forecast is performed by recursively using previous predictions to go one step ahead

Machine Learning based Forecasting

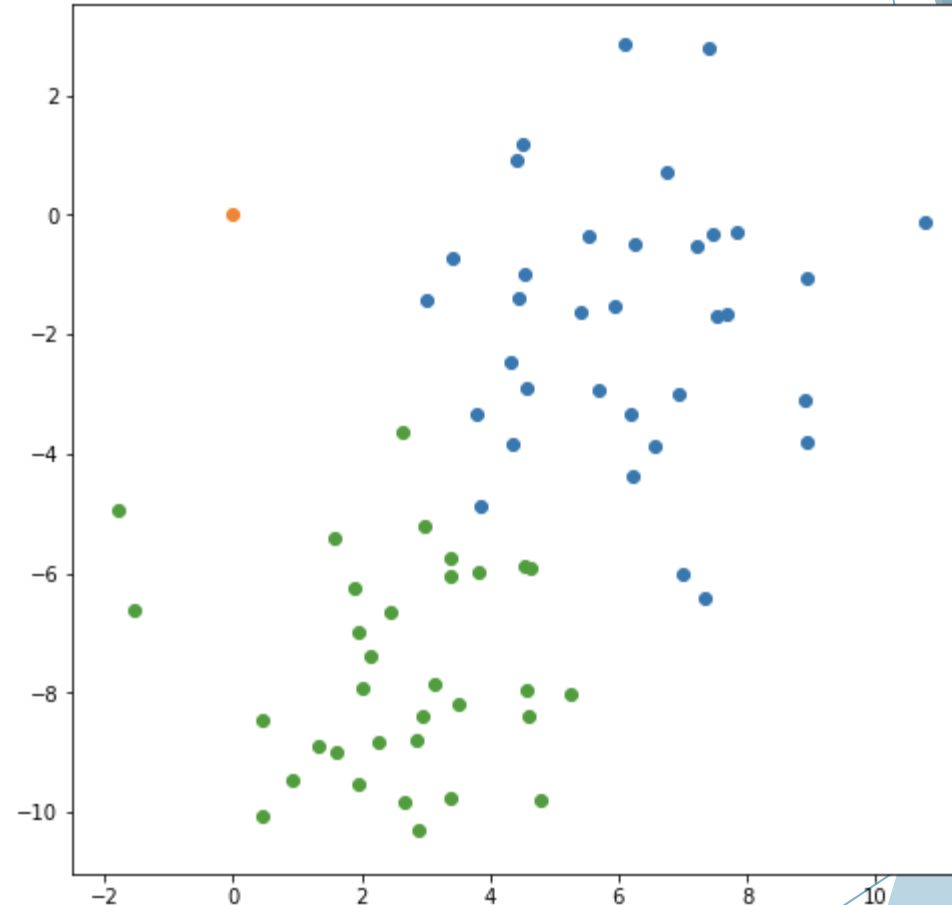
A first simple approach

- ▶ The first approach we consider is a *model-free* method
- ▶ It is based on *k-Nearest Neighbours* (kNN) algorithm - a well known "basic" ML algorithm - adapted to work on time-series/sequential data
- ▶ The basic idea behind kNN is very simple: *given an instance as input, the associated output is predicted depending on the **k most similar instances into the available dataset** (whose output values are known):*
 - ▶ *kNN for classification*
 - ▶ *kNN for regression*
 - ▶ *kNN for timeseries*

KNN for classification

Just a quick reminder...

- ▶ On the right we have a representation of the available dataset (3 classes)

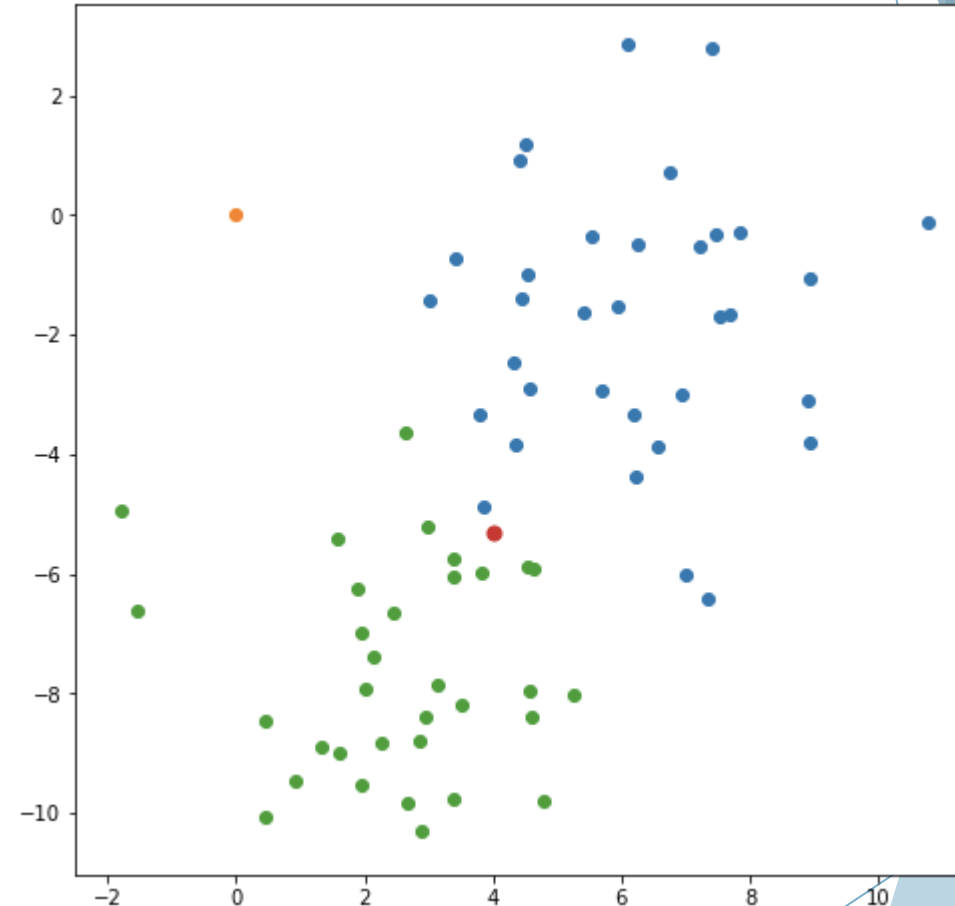


Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for classification

Just a quick reminder...

- ▶ On the right we have a representation of the available dataset (3 classes)
- ▶ We want to predict the class for the new instance (red point)

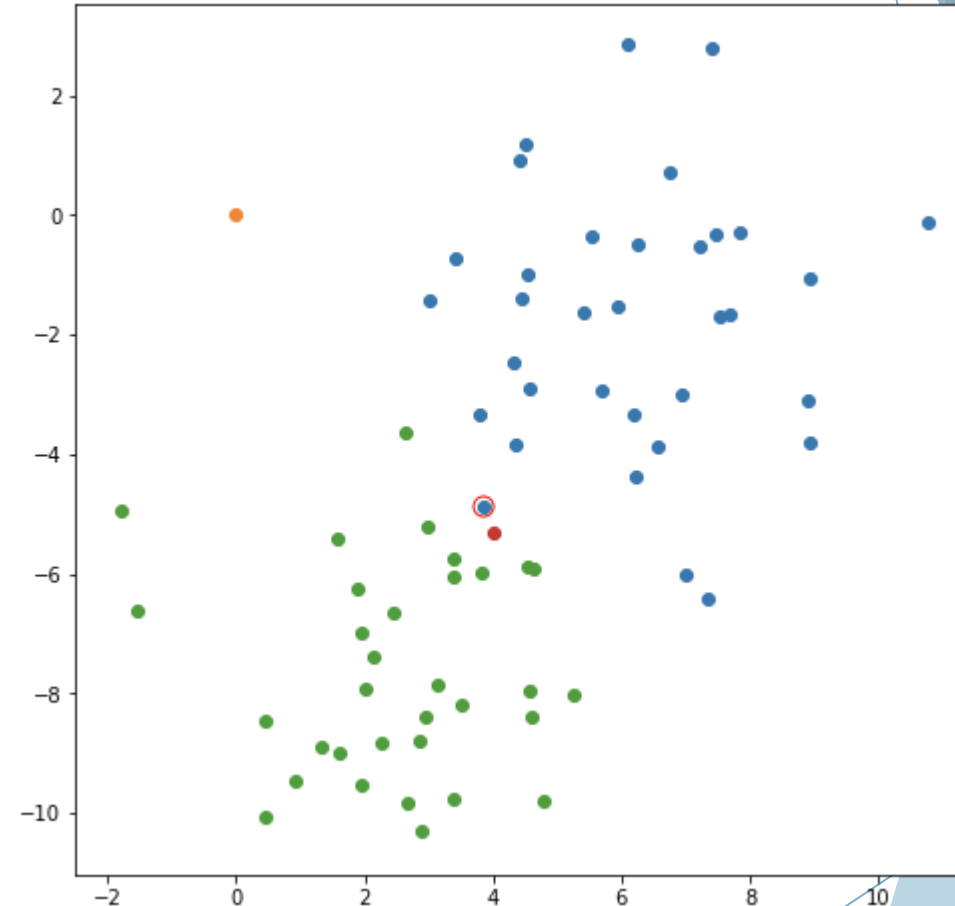


Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for classification

Just a quick reminder...

- ▶ On the right we have a representation of the available dataset (3 classes)
- ▶ We want to predict the class for the new instance (red point)
- ▶ Let's consider $k=1$ nearest neighbours
 - ▶ Predicted class is "blue"!

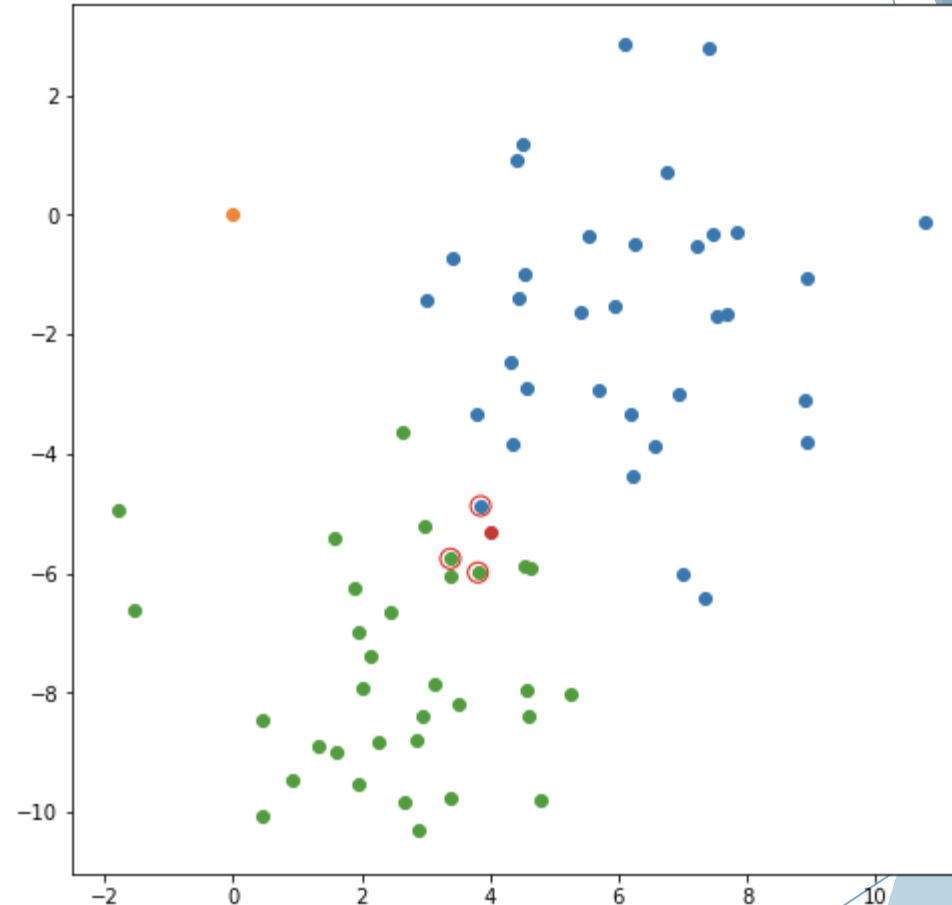


Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for classification

Just a quick reminder...

- ▶ On the right we have a representation of the available dataset (3 classes)
- ▶ We want to predict the class for the new instance (red point)
- ~~▶ Let's consider $k=1$ nearest neighbours~~
 - ~~▶ Predicted class is "blue"!~~
- ▶ Let's consider $k=3$ nearest neighbours
 - ▶ Predicted class is "green"

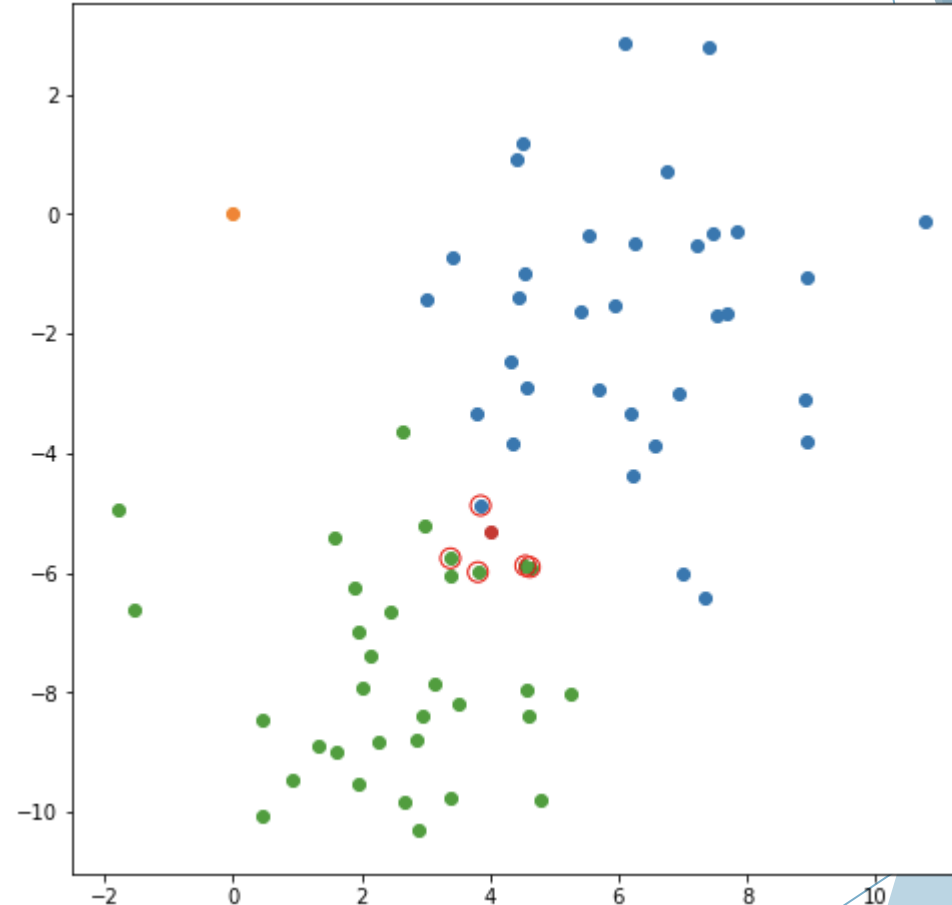


Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for classification

Just a quick reminder...

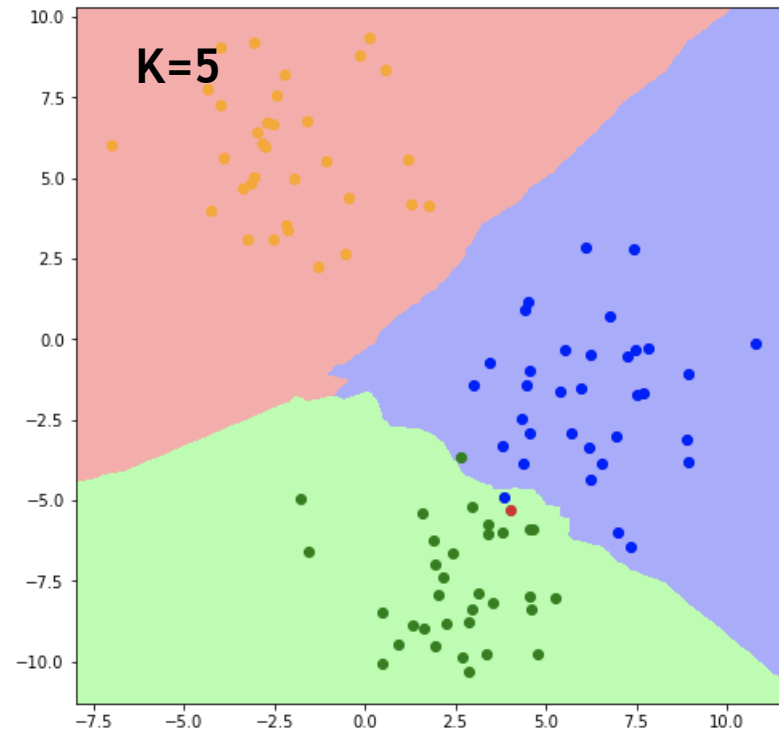
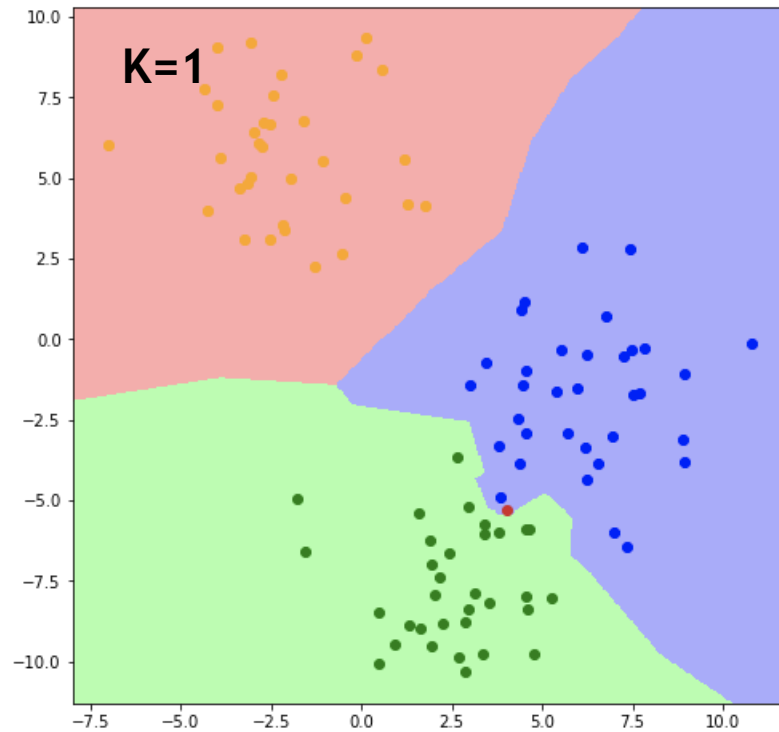
- ▶ On the right we have a representation of the available dataset (3 classes)
- ▶ We want to predict the class for the new instance (red point)
- ▶ ~~Let's consider $k=1$ nearest neighbours~~
 - ▶ ~~Predicted class is "blue"!~~
- ▶ ~~Let's consider $k=3$ nearest neighbours~~
 - ▶ ~~Predicted class is "green"~~
- ▶ Let's consider $k=5$ nearest neighbours
 - ▶ Predicted class is (again) "green"



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for classification

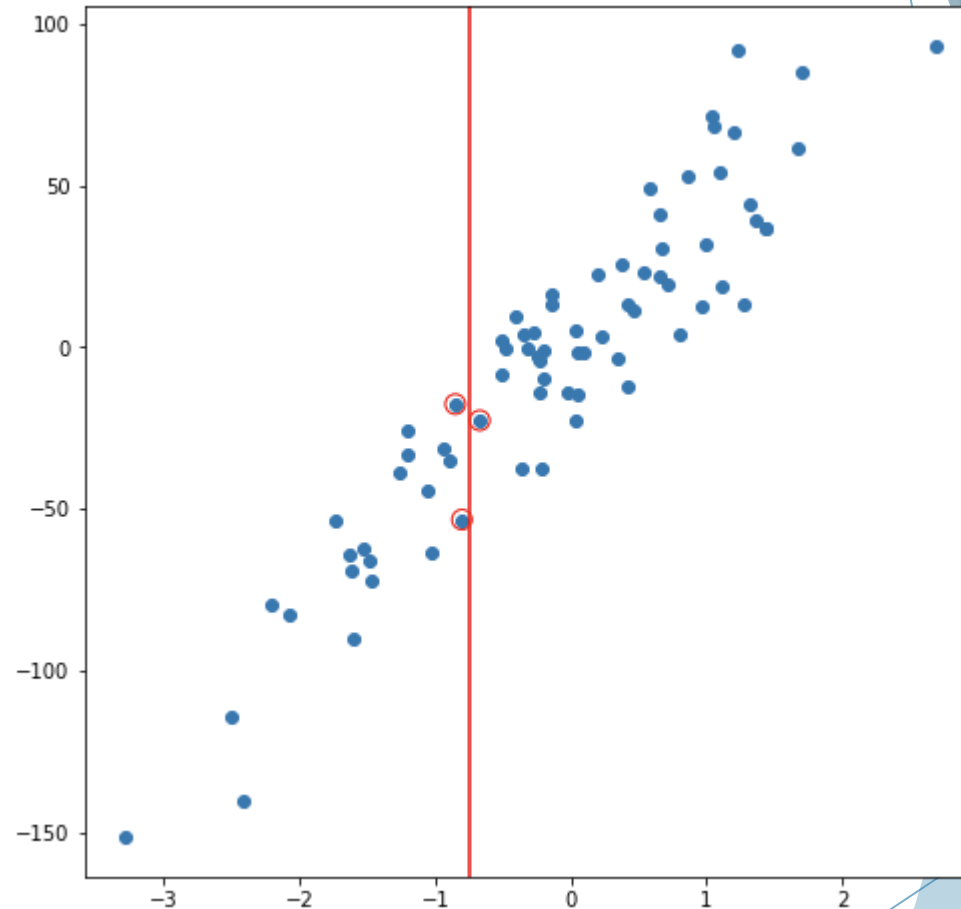
Just a quick reminder...



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for regression

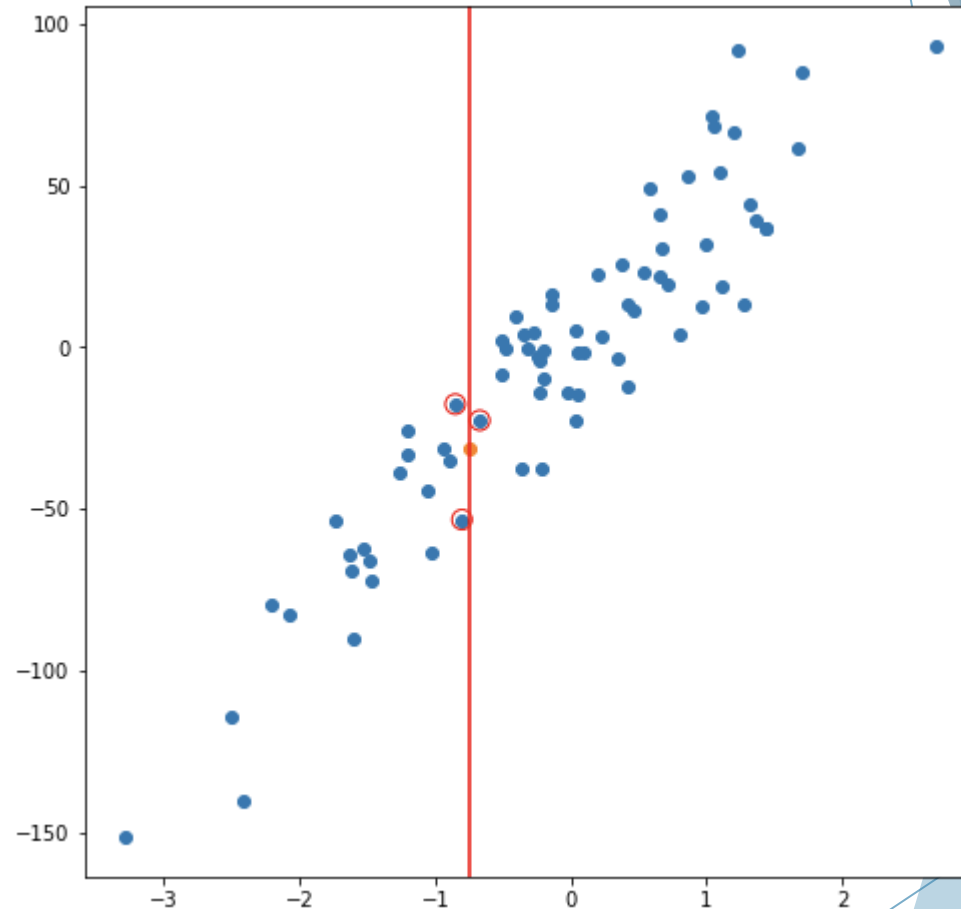
- ▶ Regression is addressed analogously:
given x , find the $k=3$ nearest neighbours



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for regression

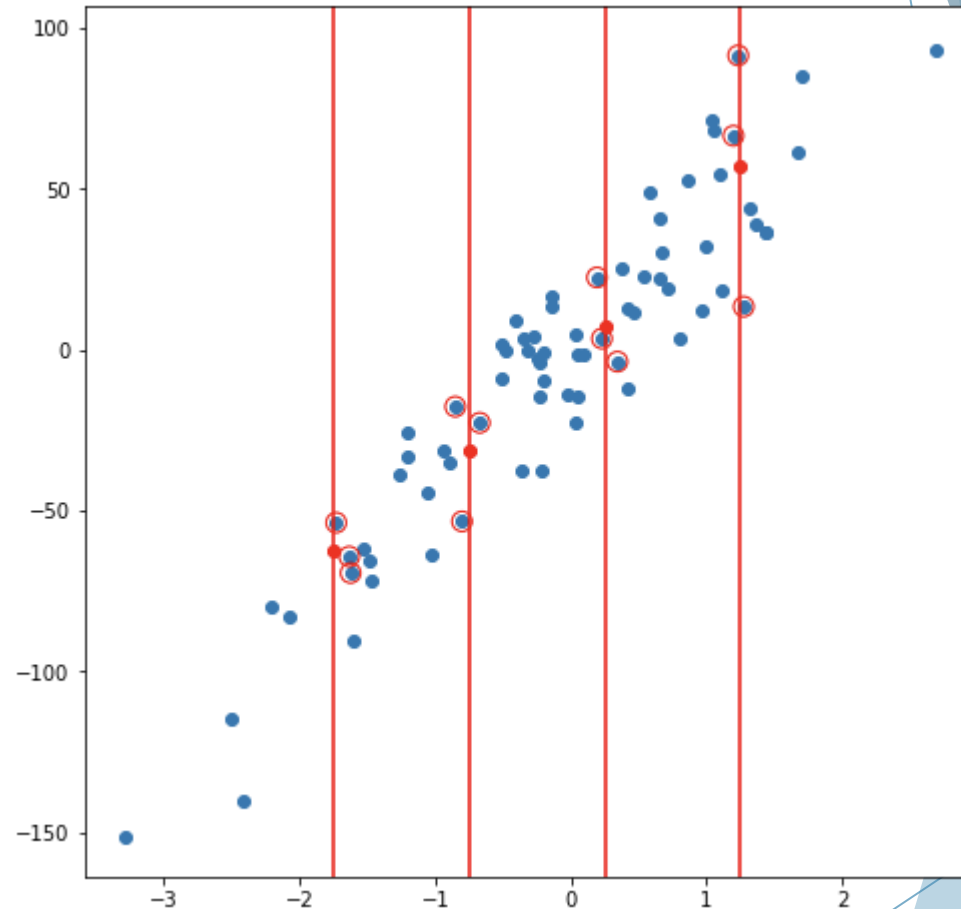
- ▶ Regression is addressed analogously:
given x , find the $k=3$ nearest neighbours
- ▶ then predict y accordingly (e.g., mean)



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for regression

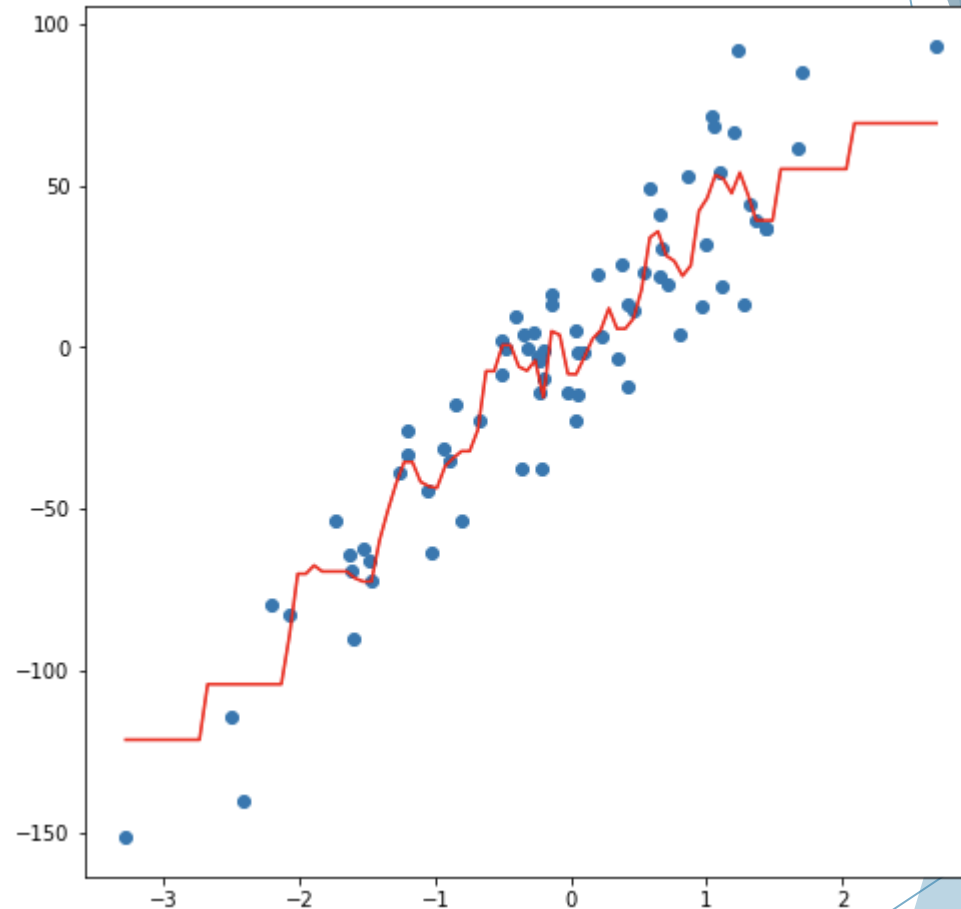
- ▶ Regression is addressed analogously:
given x , find the $k=3$ nearest neighbours
- ▶ then predict y accordingly (e.g., mean)
- ▶ Apply the same mechanism for different
values of x ...



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

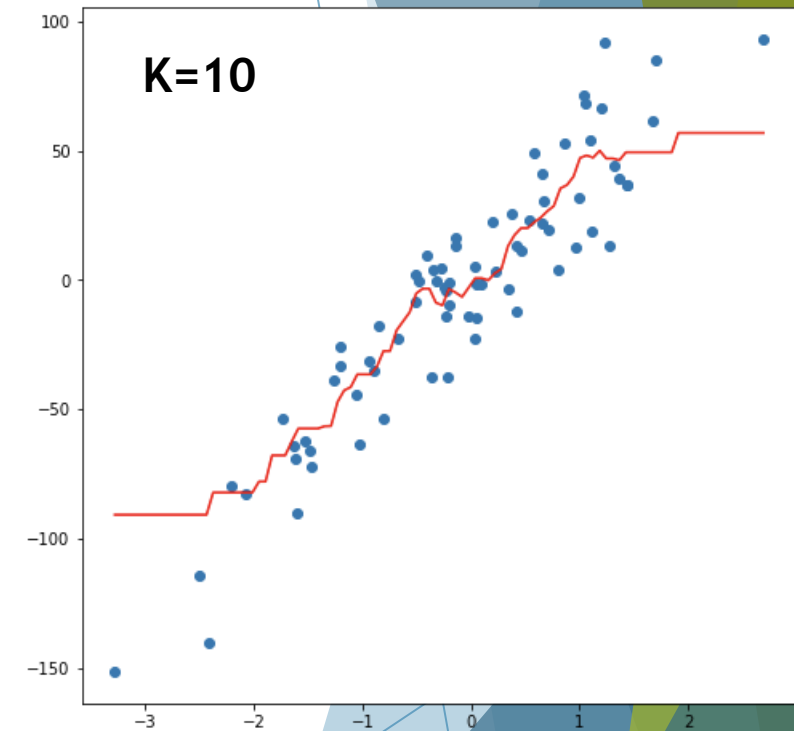
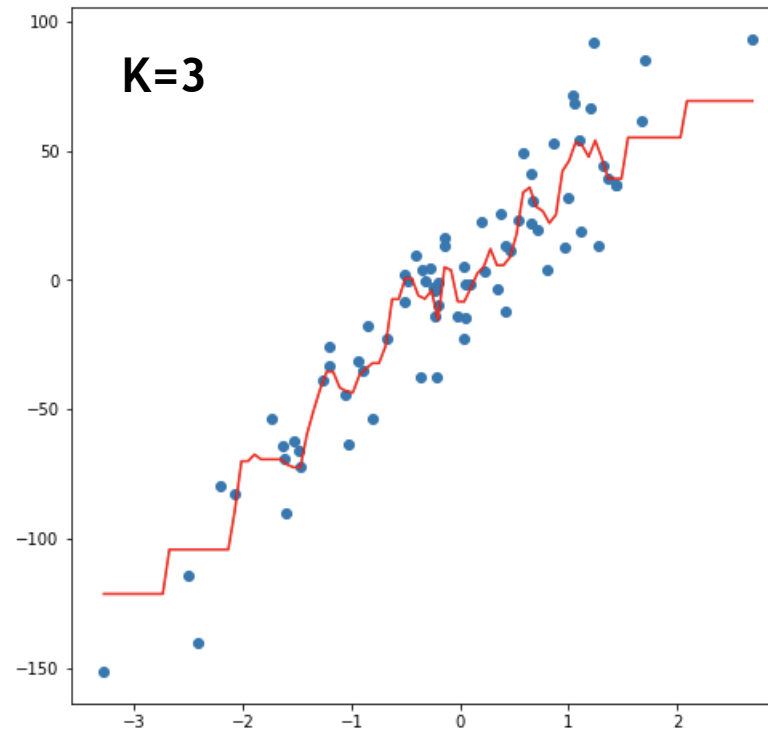
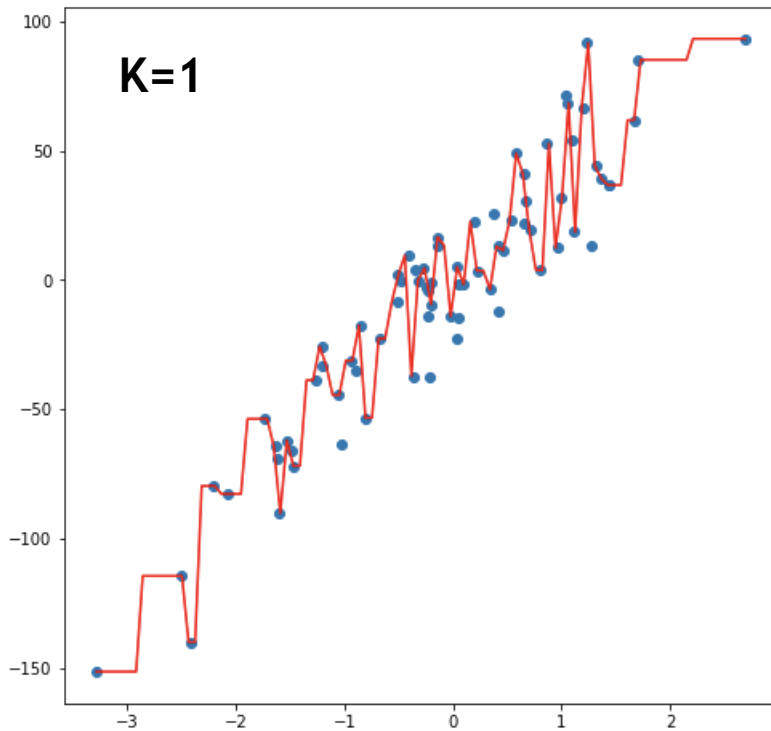
KNN for regression

- ▶ Regression is addressed analogously:
given x , find the $k=3$ nearest neighbours
- ▶ then predict y accordingly (e.g., mean)
- ▶ Apply the same mechanism for different
values of x ...
- ▶ ... leading to the regression provided by
kNN



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

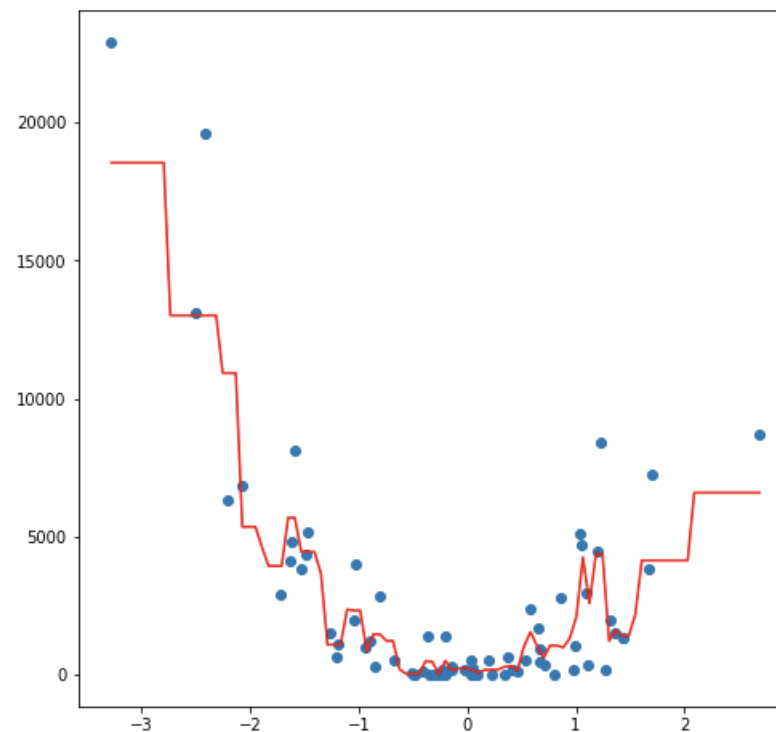
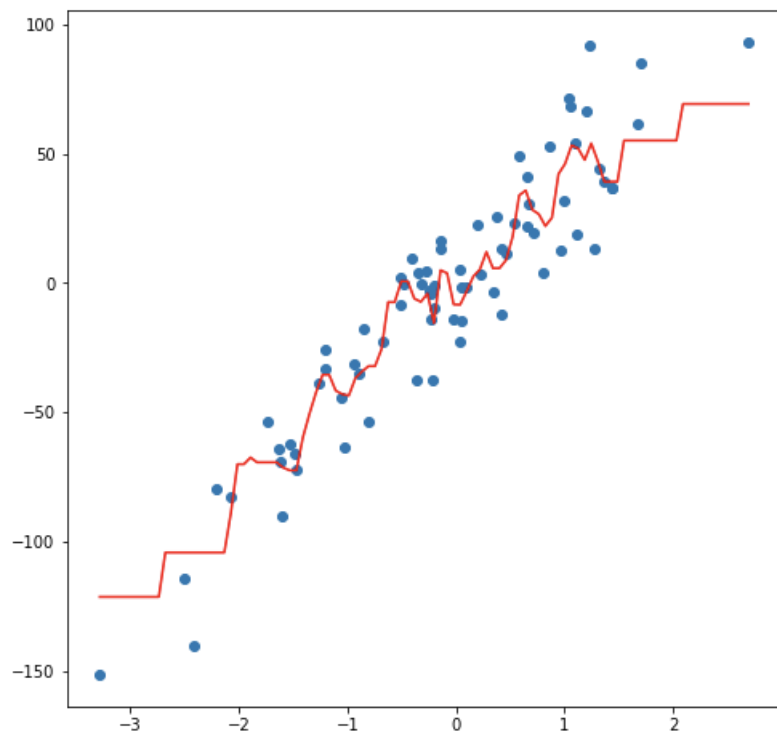
KNN for regression



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

KNN for regression

No assumptions on the non-linearities of the data: it naturally adapts to data



Source: <https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955>

Key points

- ▶ **K and smoothness** → increasing k increases smoothness
- ▶ **Distance metric** → any type of distance can be used to find the k -nearest instances
- ▶ **Combining for predicting** → different options
- ▶ **Weights** → can be used to "modulate" the contribution of each neighbour (e.g., closer instances should have a larger weight in the prediction)
- ▶ **Scaling/normalizing** → data preprocessing

- ▶ **Pros:**
 - ▶ easy to implement; naturally dealing with non-linearities, non-parametric (i.e., model-free)
- ▶ **Cons:**
 - ▶ Application on large dataset → prediction (inference) could be slow because to predict the output of a new instance KNN needs to search through the entire dataset in order to find the nearest ones
 - ▶ Lack of interpretability

KNN for time-series forecasting

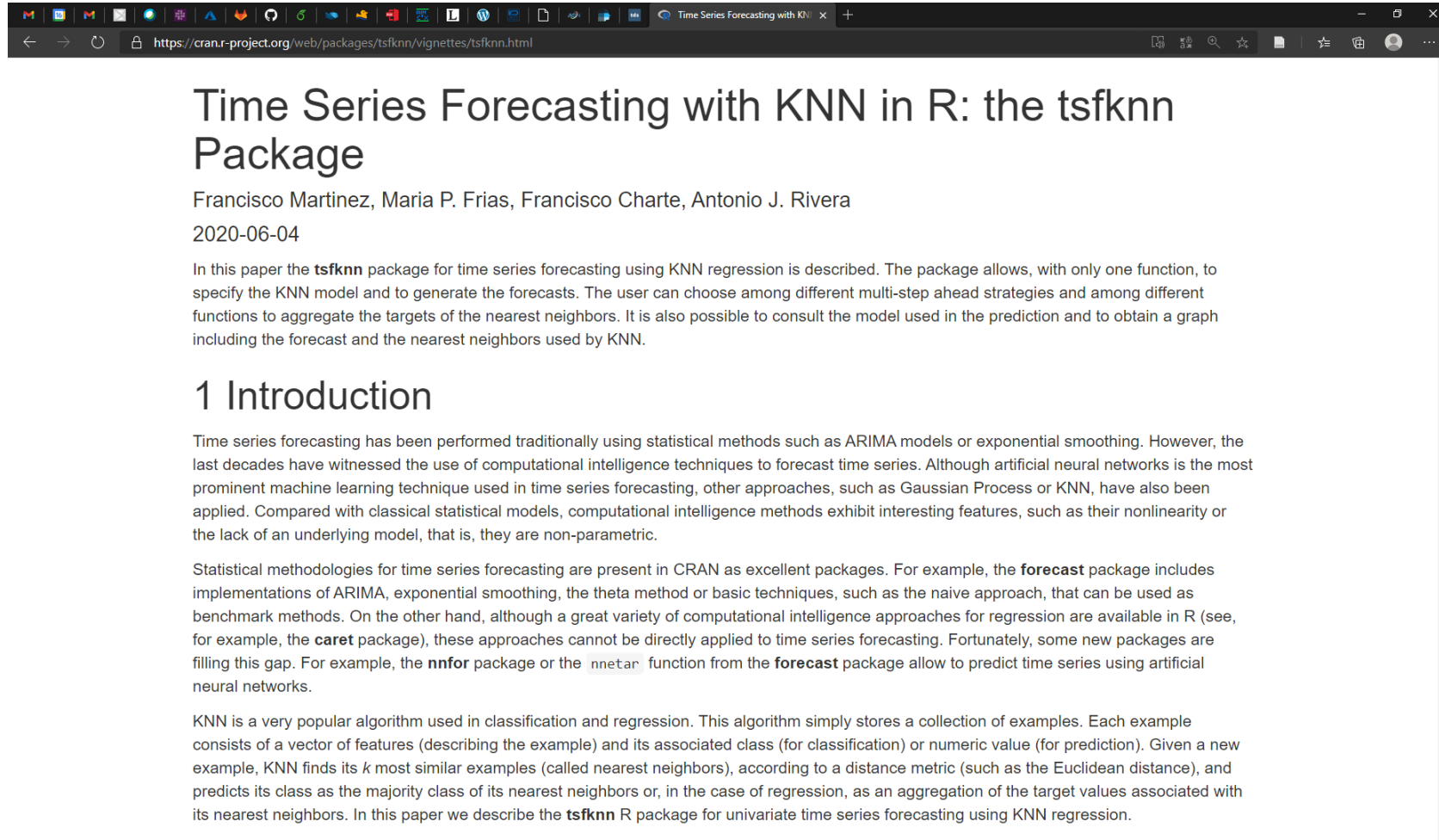
- ▶ Data is a time-series
- ▶ Distance/similarity metric must be suitable for comparing time-series
- ▶ The idea for extending KNN to time-series forecasting is quite close to *sub-sequence search*
- ▶ (At least) two hyperparameters are needed:
 - ▶ k (as in "basic" kNN algorithm)
 - ▶ p (how many past observations to consider to generate the forecast)

KNN for time-series forecasting

Basic algorithm

- ▶ Find k sub-sequences into the stream which are the "nearest" ones to the input time-series $X_{t-p:t} = \{x_i\}_{i=t-p, \dots, t}$
- ▶ Use the "future" of each one of the k sub-sequences identified to generate the forecast:
 - ▶ **Single-step** → $x_{t+h} = g\left(\{x_{t_i+h}\}_{i=1, \dots, k}\right)$ where t_i is the end of the i th subsequence among the k nearest ones, and $g(\cdot)$ can be any suitable function such as simple or weighted average
 - ▶ **Multi-step** → we want to forecast $X_{t+1:t+h}$ so $g(\cdot)$ cannot be a scalar function

tsfknn: An R package for knn based forecasting



The screenshot shows a web browser window with the address bar displaying <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>. The page title is "Time Series Forecasting with KNN in R: the tsfknn Package". The authors listed are Francisco Martinez, Maria P. Frias, Francisco Charte, and Antonio J. Rivera, with a date of 2020-06-04. The abstract states: "In this paper the **tsfknn** package for time series forecasting using KNN regression is described. The package allows, with only one function, to specify the KNN model and to generate the forecasts. The user can choose among different multi-step ahead strategies and among different functions to aggregate the targets of the nearest neighbors. It is also possible to consult the model used in the prediction and to obtain a graph including the forecast and the nearest neighbors used by KNN." The section "1 Introduction" begins with: "Time series forecasting has been performed traditionally using statistical methods such as ARIMA models or exponential smoothing. However, the last decades have witnessed the use of computational intelligence techniques to forecast time series. Although artificial neural networks is the most prominent machine learning technique used in time series forecasting, other approaches, such as Gaussian Process or KNN, have also been applied. Compared with classical statistical models, computational intelligence methods exhibit interesting features, such as their nonlinearity or the lack of an underlying model, that is, they are non-parametric." It then discusses statistical methodologies in CRAN, mentioning the **forecast** package and the **nnfor** package. Finally, it describes the KNN algorithm and the **tsfknn** R package for univariate time series forecasting using KNN regression.

Time Series Forecasting with KNN in R: the tsfknn Package

Francisco Martinez, Maria P. Frias, Francisco Charte, Antonio J. Rivera
2020-06-04

In this paper the **tsfknn** package for time series forecasting using KNN regression is described. The package allows, with only one function, to specify the KNN model and to generate the forecasts. The user can choose among different multi-step ahead strategies and among different functions to aggregate the targets of the nearest neighbors. It is also possible to consult the model used in the prediction and to obtain a graph including the forecast and the nearest neighbors used by KNN.

1 Introduction

Time series forecasting has been performed traditionally using statistical methods such as ARIMA models or exponential smoothing. However, the last decades have witnessed the use of computational intelligence techniques to forecast time series. Although artificial neural networks is the most prominent machine learning technique used in time series forecasting, other approaches, such as Gaussian Process or KNN, have also been applied. Compared with classical statistical models, computational intelligence methods exhibit interesting features, such as their nonlinearity or the lack of an underlying model, that is, they are non-parametric.

Statistical methodologies for time series forecasting are present in CRAN as excellent packages. For example, the **forecast** package includes implementations of ARIMA, exponential smoothing, the theta method or basic techniques, such as the naive approach, that can be used as benchmark methods. On the other hand, although a great variety of computational intelligence approaches for regression are available in R (see, for example, the **caret** package), these approaches cannot be directly applied to time series forecasting. Fortunately, some new packages are filling this gap. For example, the **nnfor** package or the `nnetar` function from the **forecast** package allow to predict time series using artificial neural networks.

KNN is a very popular algorithm used in classification and regression. This algorithm simply stores a collection of examples. Each example consists of a vector of features (describing the example) and its associated class (for classification) or numeric value (for prediction). Given a new example, KNN finds its k most similar examples (called nearest neighbors), according to a distance metric (such as the Euclidean distance), and predicts its class as the majority class of its nearest neighbors or, in the case of regression, as an aggregation of the target values associated with its nearest neighbors. In this paper we describe the **tsfknn** R package for univariate time series forecasting using KNN regression.

tsfknn: An R package for knn based forecasting

- You can find the paper at the following link:
<https://journal.r-project.org/archive/2019/RJ-2019-004/RJ-2019-004.pdf>

Time Series Forecasting with KNN in R: the tsfknn Package

by Francisco Martínez, María P. Frías, Francisco Charle and Antonio J. Rivera

Abstract In this paper the `tsfknn` package for time series forecasting using k -nearest neighbor regression is described. This package allows users to specify a KNN model and to generate its forecasts. The user can choose among different multi-step ahead strategies and among different functions to aggregate the targets of the nearest neighbors. It is also possible to assess the forecast accuracy of the KNN model.

Introduction

Time series forecasting has been performed traditionally using statistical methods such as ARIMA models or exponential smoothing (Hyndman and Athanasopoulos, 2014). However, the last decades have witnessed the use of computational intelligence techniques to forecast time series. Although artificial neural networks is the most prominent machine learning technique used in time series forecasting (Zhang et al., 1998), other approaches, such as Gaussian Processes (Andrawis et al., 2011) or KNN (Martínez et al., 2017), have also been applied. Compared with classical statistical models, computational intelligence methods exhibit interesting features, such as their nonlinearity or the lack of an underlying model, that is, they are non-parametric.

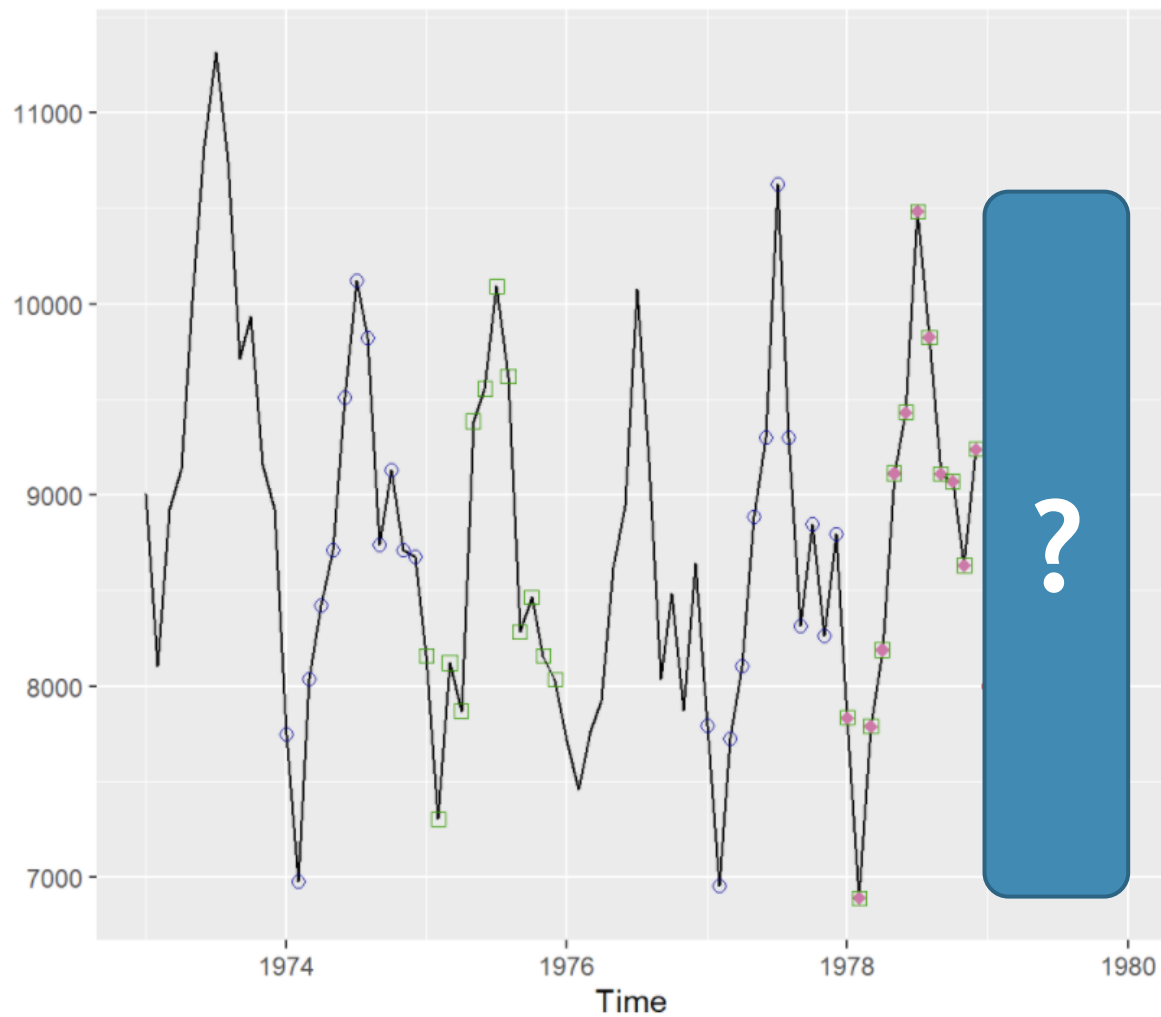
Statistical methodologies for time series forecasting are present in R as excellent packages. For example, the `forecast` package (Hyndman and Khandakar, 2008) includes implementations of ARIMA, exponential smoothing, the Theta method (Hyndman and Billah, 2003) or basic techniques that can be used as benchmark methods—such as the random walk approach. On the other hand, although a great variety of computational intelligence approaches for regression are available in R, such as the `caret` package (Kuhn, 2008), these approaches cannot be directly applied to time series forecasting. Fortunately, some new packages are filling this gap. For example, the `nnfor` package (Kourentzes, 2017) or the `nnetar` function from the `forecast` package allow users to predict time series using artificial neural networks.

KNN is a very popular algorithm used in classification and regression (Wu et al., 2007). This algorithm simply stores a collection of examples. In regression, each example consists of a vector of features describing the example and its associated numeric target value. Given a new example, KNN finds its k most similar examples, called nearest neighbors, according to a distance metric such as the Euclidean distance, and predicts its value as an aggregation of the target values associated

tsfknn

- ▶ Both **single-step** & **multi-step**
- ▶ For multi-step, both **multi-output** & **recursive**
- ▶ Easy-to-use and extend

Multi-output forecast

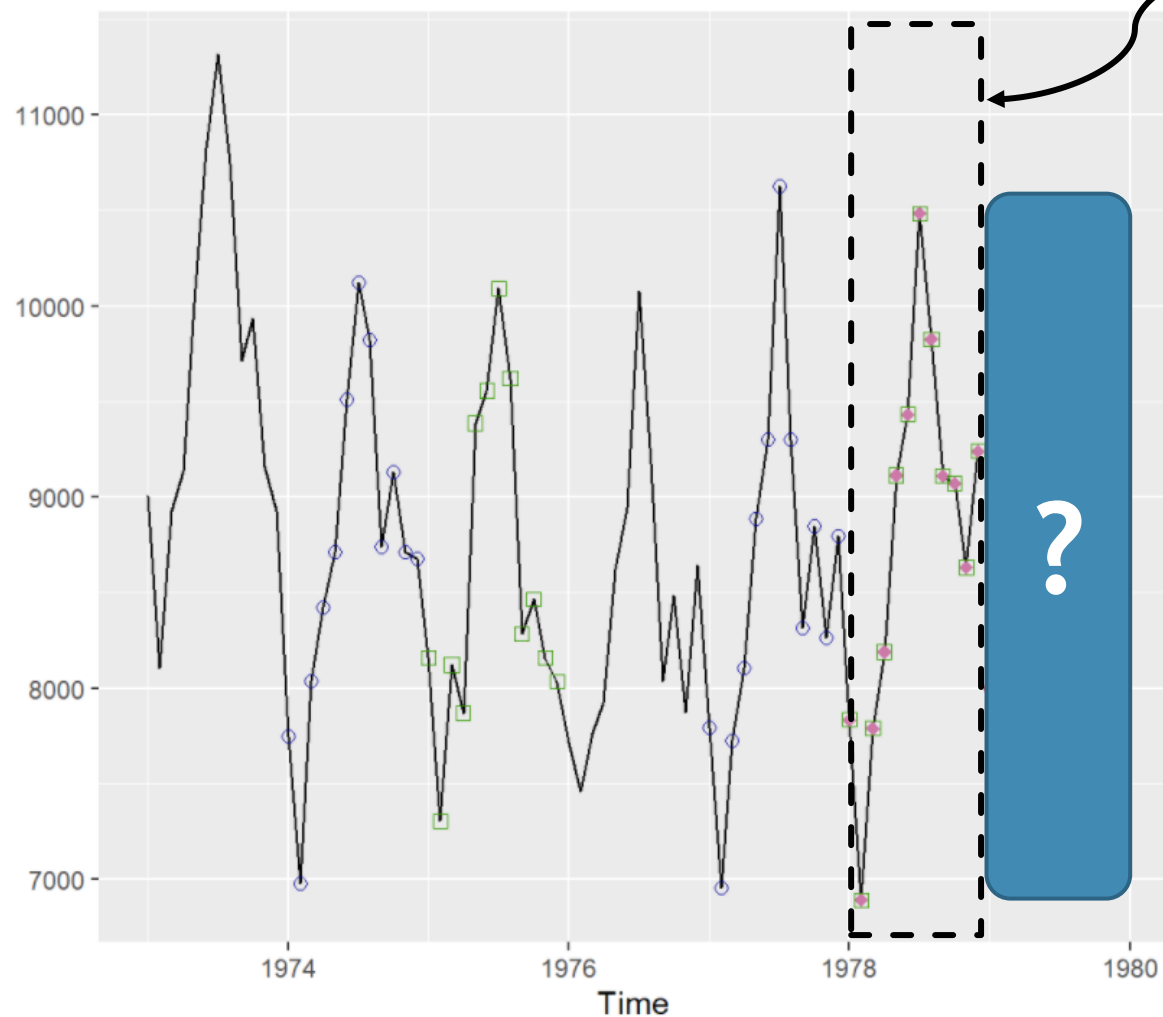


Source: <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>

Data point

- NN Features
- NN Targets
- ◆ Instance
- Forecast

Multi-output forecast



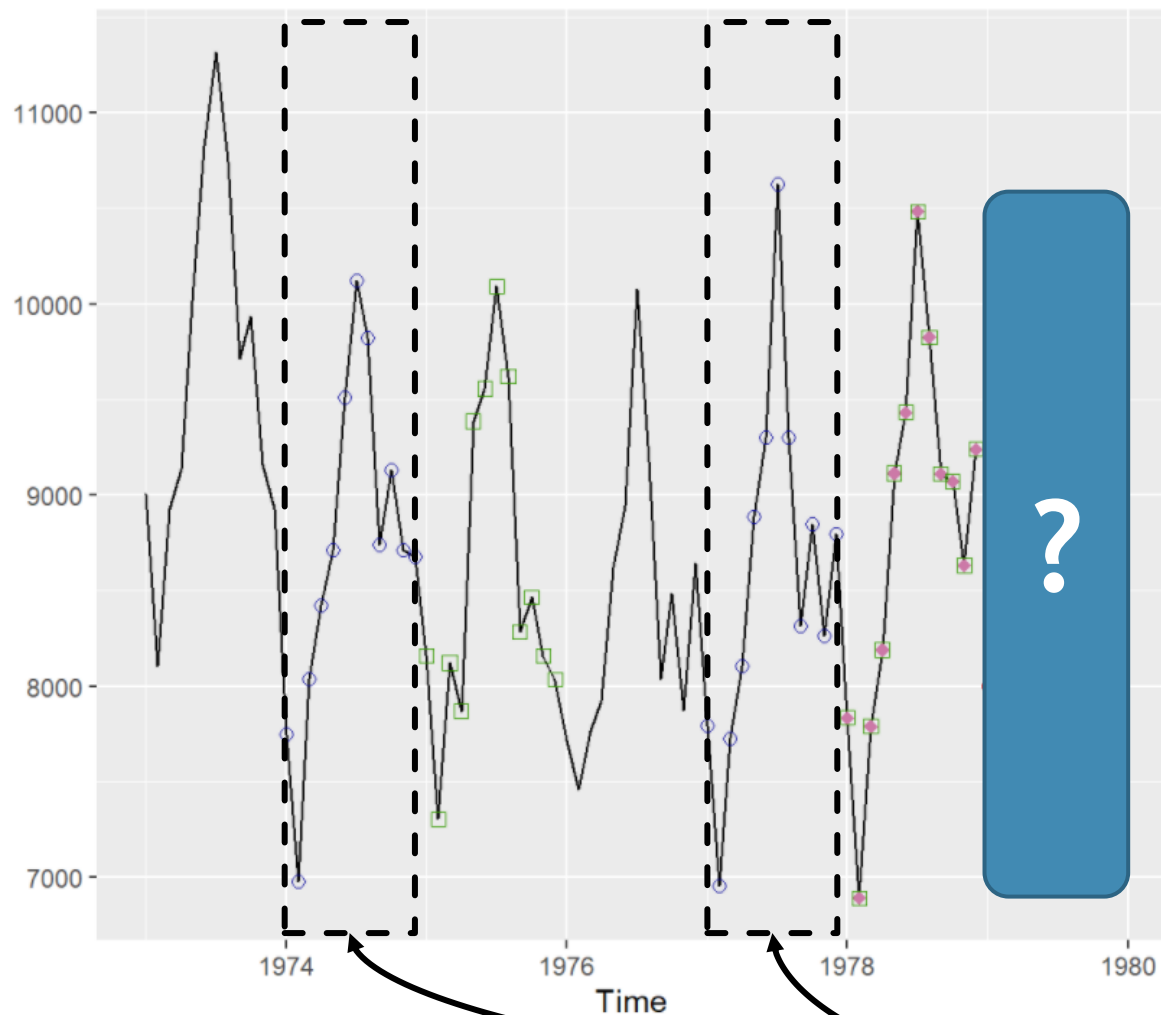
p - length of the 'history'

Source: <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>

Data point

- NN Features
- NN Targets
- ◆ Instance
- Forecast

Multi-output forecast

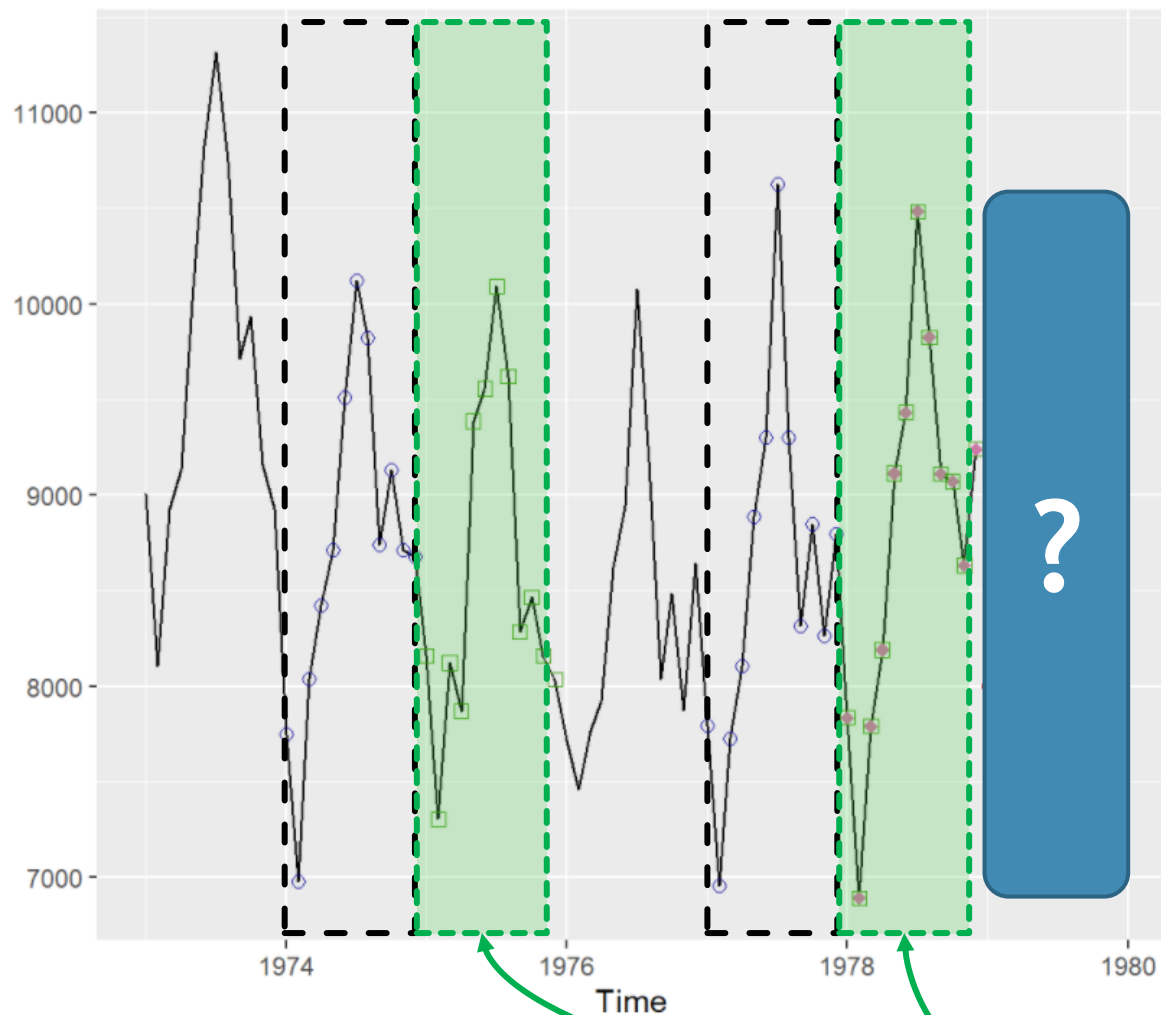


Source: <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>

Business Intelligence per i Servizi Finanziari 2023/2024 - Candelieri A.

k - the number of neighbours

Multi-output forecast



Source: <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>

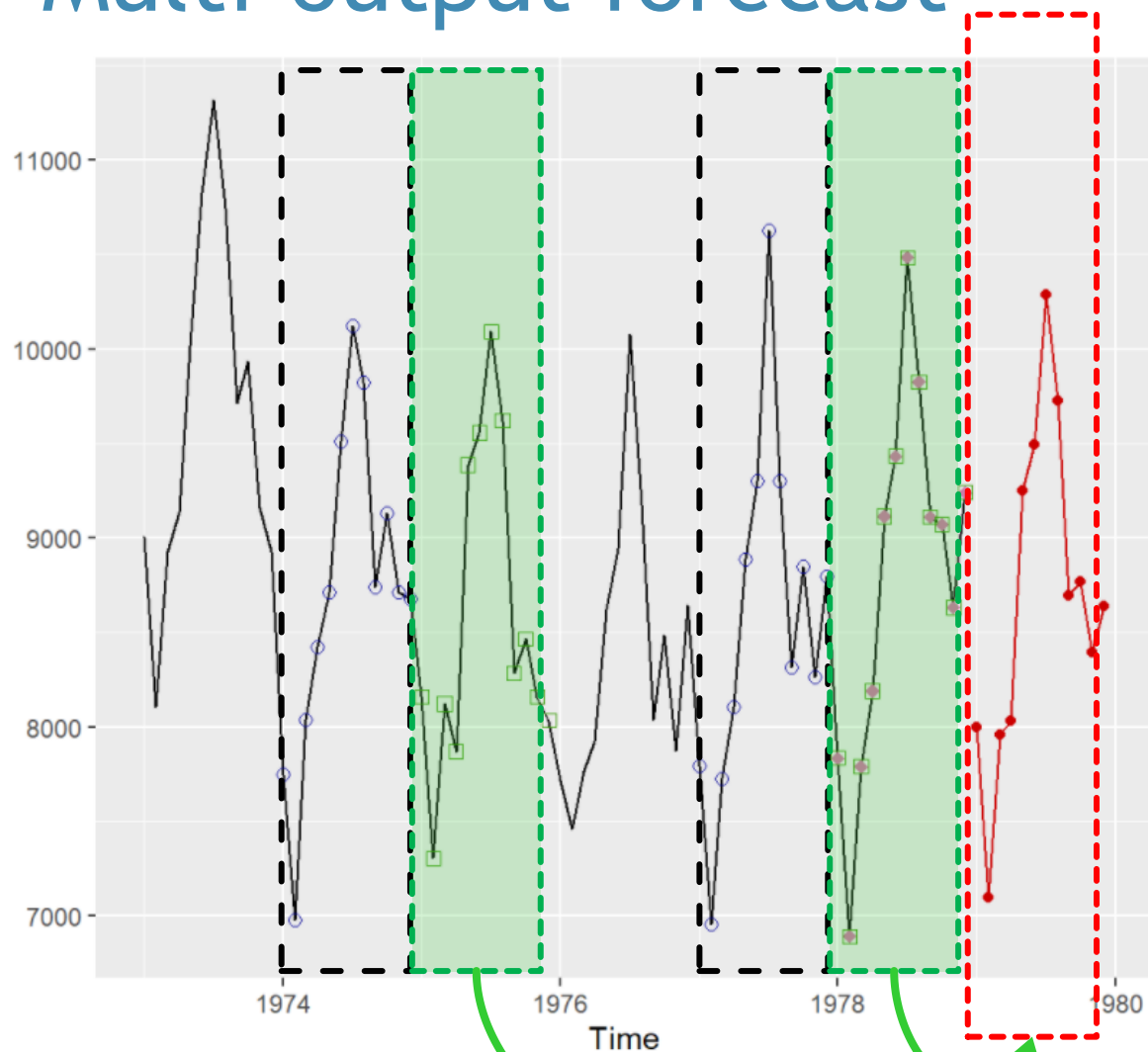
Data point

- NN Features
- NN Targets
- ◆ Instance
- Forecast

Business Intelligence per i Servizi Finanziari 2023/2024 - Candelieri A.

possible futures

Multi-output forecast



Source: <https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html>

Data point

- ◇ NN Features
- NN Targets
- ◆ Instance
- Forecast

Some issues/limitations

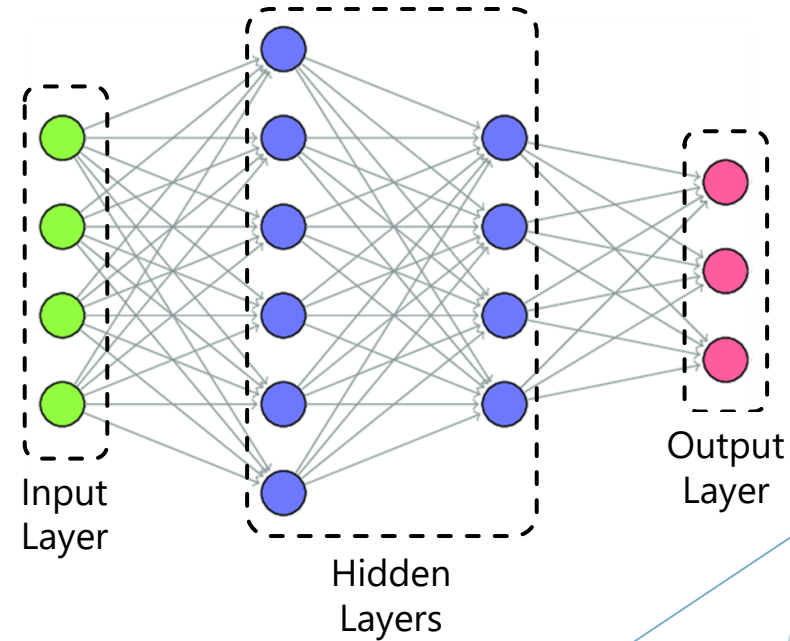
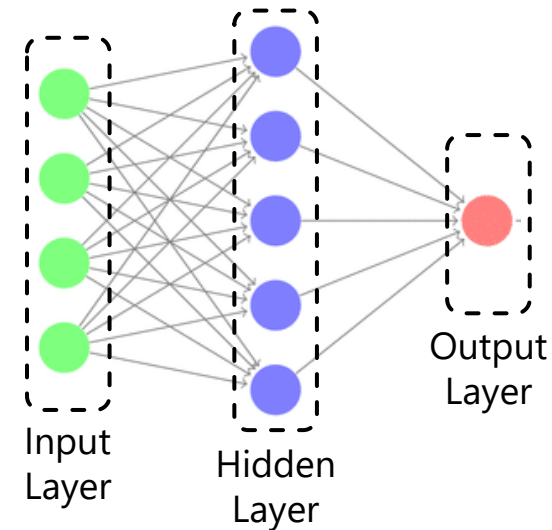
- ▶ Overlapping or non-overlapping neighbours?
- ▶ Length of the "history" (i.e., p) → the larger the history, the lower the maximum number of possible neighbours
- ▶ Length of the horizon (specifically in multi-output forecasting)
- ▶ Distance metric (tsfknn uses Euclidean, but you can implement your distance function)
- ▶ Generating the forecast (tsfknn implements mean, median and weighted mean)
- ▶ Preprocessing



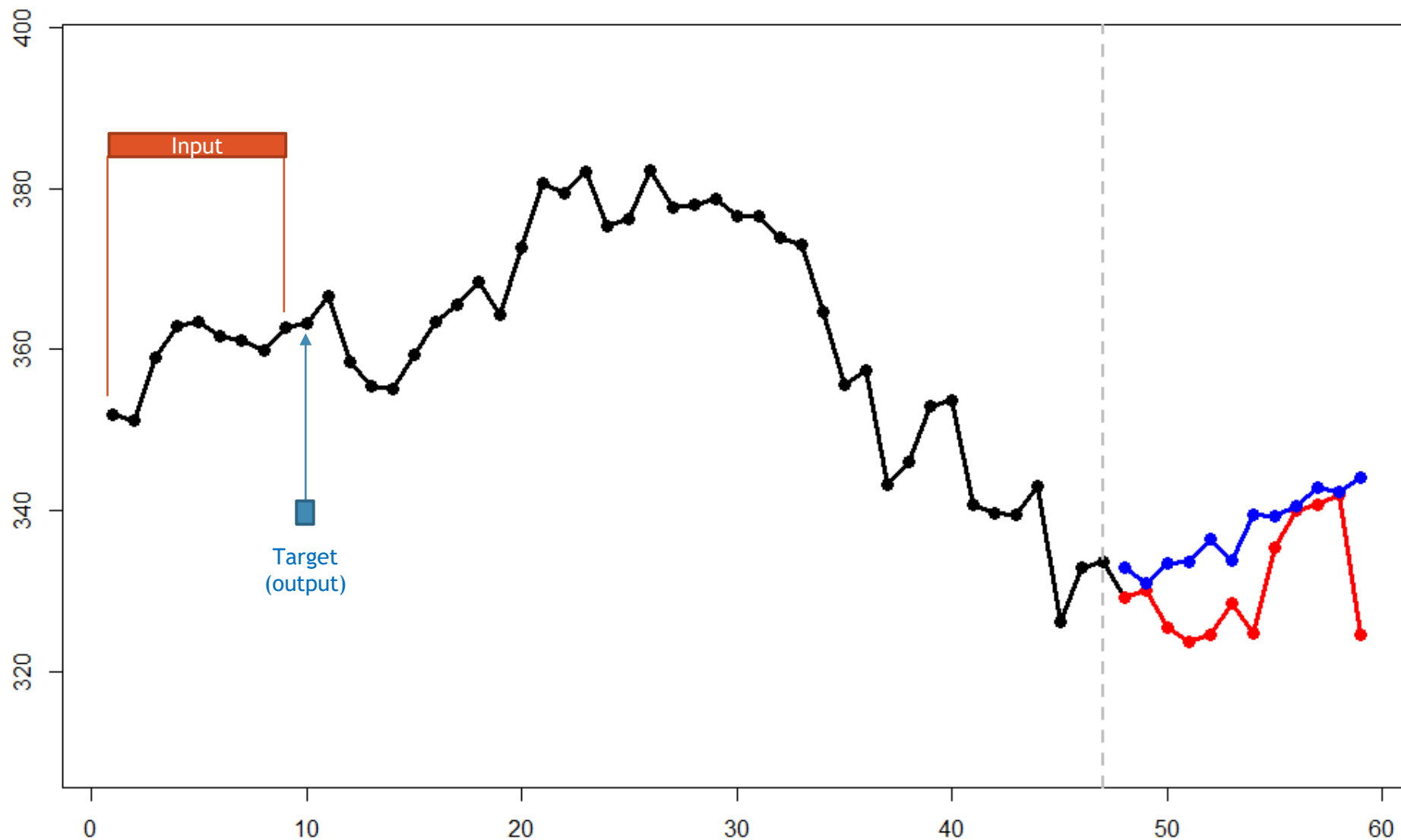
Come back to ANN for
multi-step prediction

Layer

- ▶ The power of NNs lay in the possibility to combine a multitude of computational units (neurons)
- ▶ A layer is the conceptual element to organize neurons within a (feedforward) NN
- ▶ **Input Layer**
 - ▶ Is the set of input neurons
- ▶ **Hidden Layer(s)**
 - ▶ A Hidden Layer is a set of neurons performing the computational "core" of an NN
- ▶ **Output Layer**
 - ▶ Is the set of neurons which combine the output of the hidden layer in the final output
- ▶ Size of both Input and Output layer are given by the task

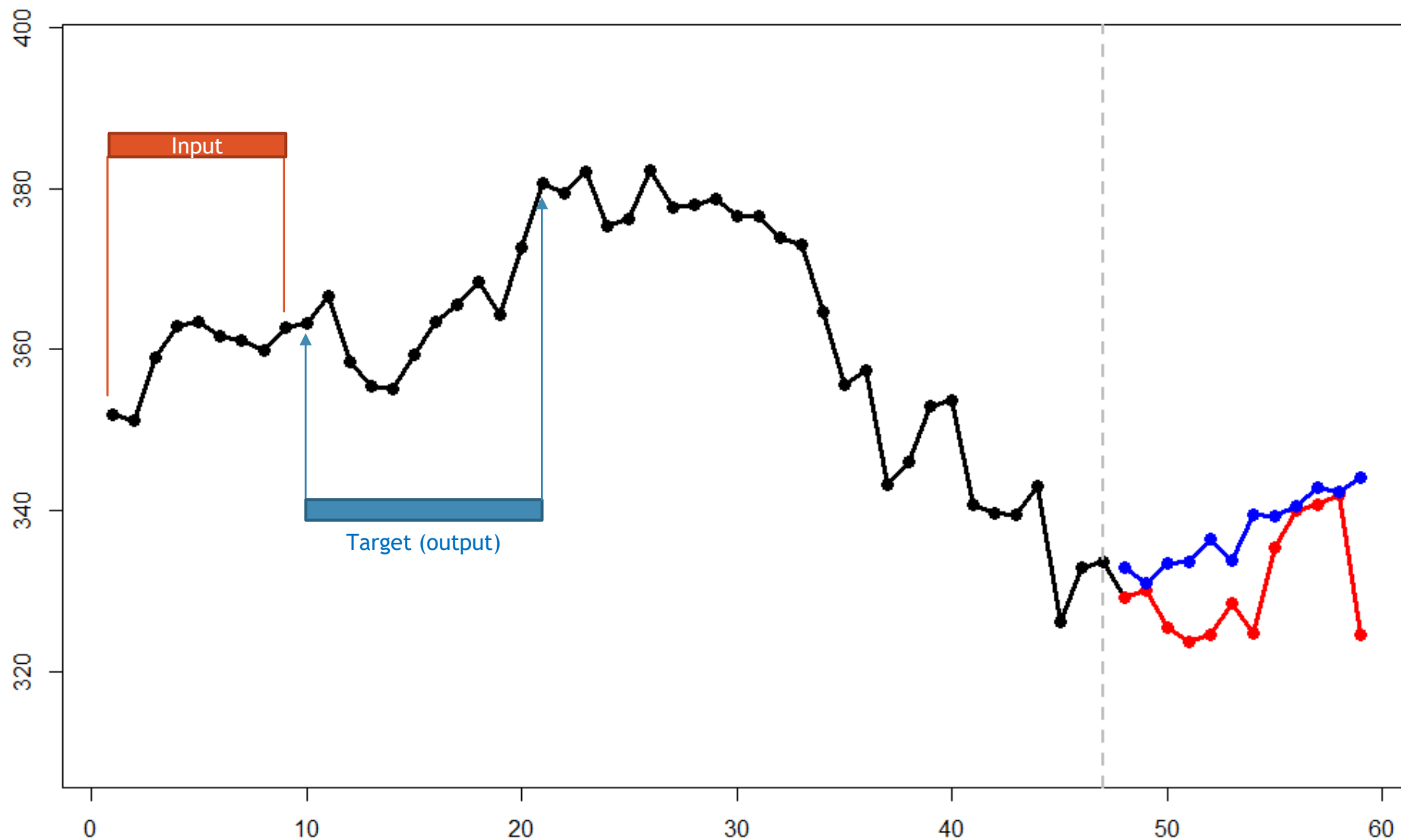


Do you remember the single-step case?



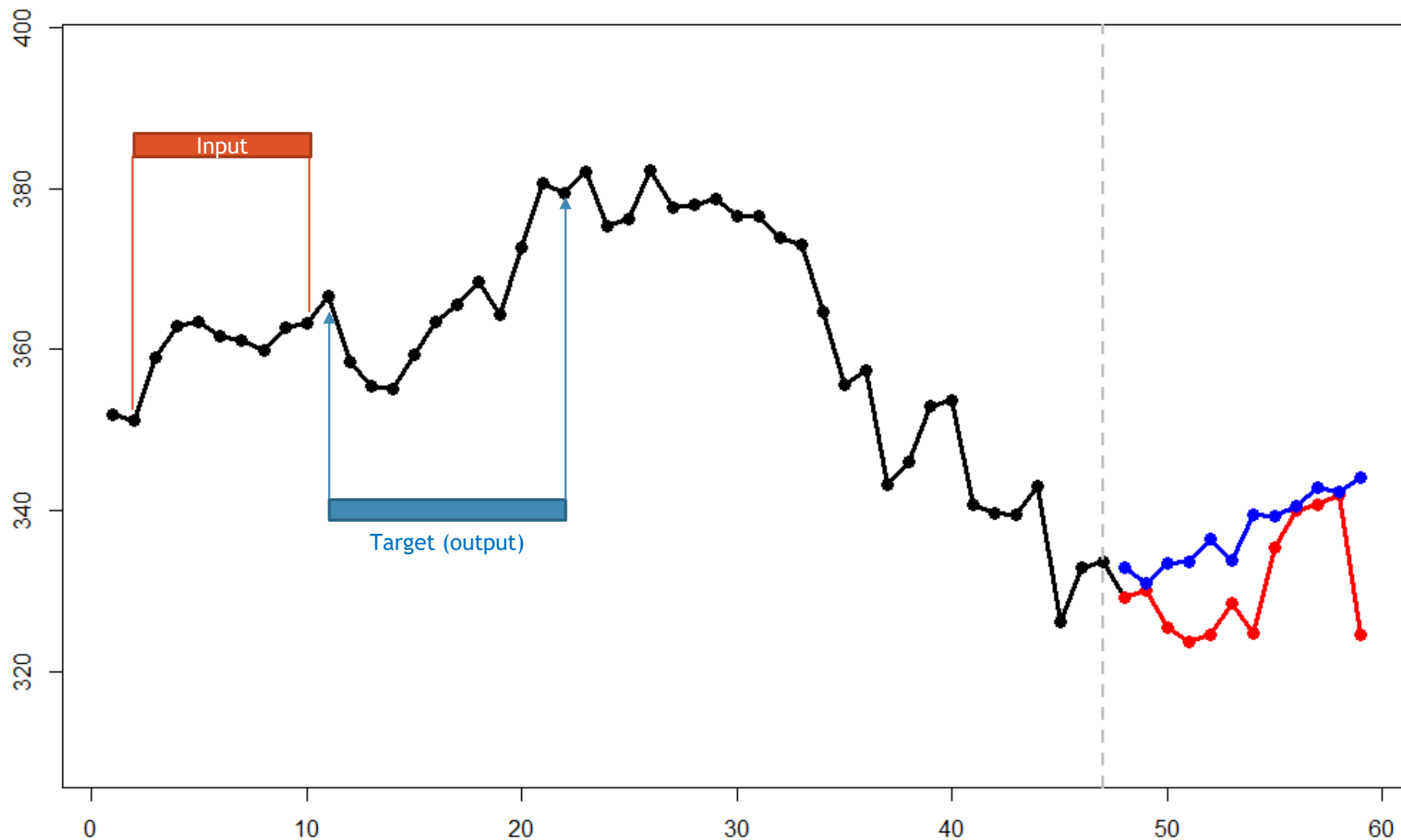
First data

What happens in the multi-output case?



First data

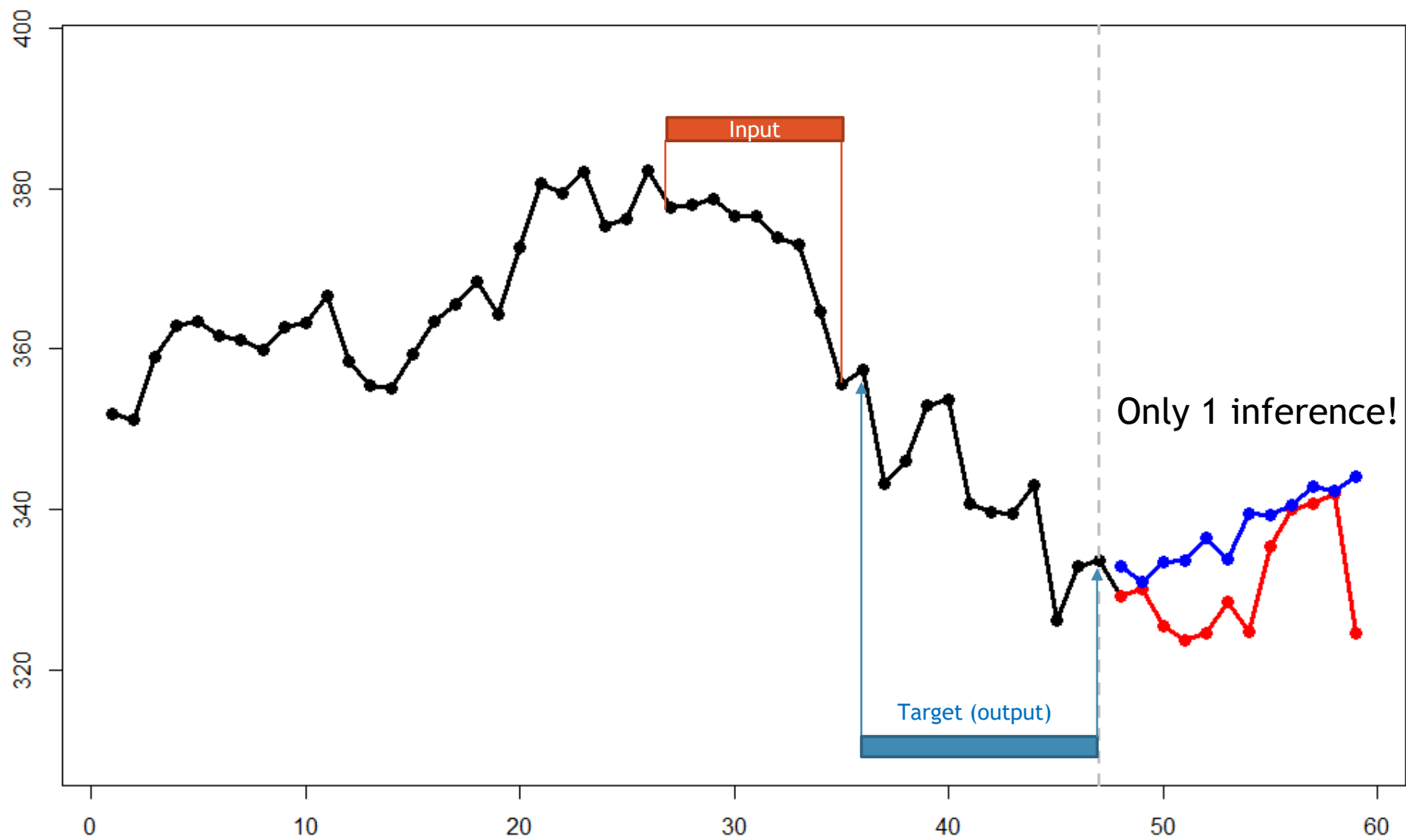
What happens in the multi-output case?



First data

Second data

What happens in the multi-output case?



First data	
Second data	
last data	

Important

- ▶ Setting the value of p (i.e., length of the history) and h (i.e., length of the horizon):
 - ▶ depends on the size of available dataset!
 - ▶ impacts on the ANN's architecture
 - ▶ impacts on (training) computational time